

MasterLINK®

Meaningful **I**nformation for the entire **W**orkforce

MasterLink Corporation

3649 All American Blvd., Orlando, FL 32810-4726

407/299-3900 • Fax 407/299-8200 • E-mail: atek@gdi.net

“Confidential”

December 18, 1998

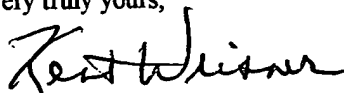
Mr. David Kershaw
Program Manager
Technical Research Development Authority
Investment Initiative for Energy Technologies
6750 South Highway U. S 1
Titusville, FL 323780

Dear Dave:

The information you requested from us on this date is attached. It is comprised of three major elements. A description of each element, along with its location within the document, is included on the following pages. Please call me if you have any questions or comments.

We are looking forward to talking with you again soon.

Very truly yours,



Kent A. Weisner
MasterLink Corporation

Exhibit 3

Use Cases (pages 4-8): The MasterLink functional requirements are captured in a series of Use Cases. Use Cases are an OO standard way to capture the “system operations”; that is, the series of interactions with a user (a mobile worker, a manager, another computer system) of the MasterLink system. These interactions define the system’s capabilities; they are the equivalent of function points in a structured programming methodology.

Design (pages 9-16): The MasterLink design is captured in the Use Case Schemas and the definition of the Agents. There is one Use Case schema for each Use Case. The schema refines the requirements at the use case level to a series of steps. Each step contributes to the functionality encapsulated in the Use Case, and includes the definition of any interactions with the object model and agents that are required to complete the step.

The “**System Architecture Overview**” (page 17) shows the positioning of the intelligent agents in the overall MasterLink Architecture. The Agents are designed as a set of Enterprise Java Beans, running on top of CORBA. We will use Java Bean’s ability to run RMI (the Java inter-process communication protocol built into the language’s syntax and semantics) over IIOP (CORBA’s inter-ORB communication protocol). This will allow us to develop with native Java (a rapid development environment) while retaining the ability to be called by or call to routines/objects/components designed in other languages and environments that conform to the CORBA protocols. Agents communicate using standard RMI-like function calls. The contents of each call is a series of expressions defined by KQML-like semantics.

The “**Physical Architecture for a Distributed Intelligent Work Management System**”(page 18) shows the layering of the system middleware. The agents (Enterprise Java Beans) are layered over IIOP, that is, over an ORB, which runs over TCP/IP. Additionally, the client workstations (mobile workstations on hand-held units, manager workstations on desktops) run HTTP over TCP/IP. For example, the workstations can run over a corporate intranet. Each conversation would begin with the user submitting normal HTTP to the web server. The web server would then use various CGI-compatible mechanisms, such as a servlet, to mediate between the client’s HTTP world and the business server’s CORBA world. That is, the servlet, which is part of the HTTP server, is a CORBA client, and uses Java/RMI/IIOP to communicate with the servers (e.g., with the business objects and the intelligent agents).

Each agent is built using off the shelf (COTS software). The most complicated is the scheduler, which performs near-real time scheduling and rescheduling. It uses a classic constraint-based scheduler, which is embodied in the Scheduler product that we will purchase and embed in the scheduler agent. The product that we have chosen for this purpose is the ILOG scheduler, a constraint-based scheduler that runs on top of the well respected ILOG Solver engine.

“**Job State Transitions**” (page 19) shows the other use of AI technology: the use of rules to define the progression of states that tasks go through. A commercial expert system shell that does standard forward chaining will be embedded in this agent and used to define these rules.

Domain Model (pages 20-25) : The MasterLink domain model (and, following the “Golden Rule of OO” the high level object model) is captured in a series of UML drawings made with Rational Rose and captured in Rose Petal files. These drawings capture the general mobile work domain (targets, tasks, etc.) as well as the Facilities-specific Construction Specification Institute code (the ontology for the world of Facilities Management). This model will be refactored during the development period due to considerations of extensibility and performance requirements.

MasterLink System Charter Statement

CONFIDENTIAL

The MasterLink Facilities Intelligent Work Management Technology will be able to:

- Describe all operational and maintenance requirements of a target (anything that needs work) by utilizing, or creating, industry standard definitions.
- Optimize the skill delivery of internal or external resources needed to fulfill target requirements.
- Automate the supervisory processes of work planning, scheduling, and dispatching.
- Enable corporate management to implement, and monitor the results of, policy through the use of templates. Policies are the rules controlling the requirements of targets, and resources performing work.
- Support mobile worker resources with all job information when they need it.
- Automate field data reporting/collection.
- Easily adapt to other industries, in addition to facilities management.

MasterLink System Function Groups and Associated Use Cases

SYSTEM FUNCTION GROUP: System Initiation

USE CASE TITLE Maintain JobTypes

DESC: Gives context to Job(s), e.g., Planned Maintenance Job.

RESULTS: A method of classifying Jobs.

USE CASE TITLE Maintain Resource

DESC: A resource is a human used to accomplish work, e.g., a manager, a mechanic, etc..

RESULTS: A method of classifying resources.

USE CASE TITLE Maintain Target

DESC: Input information about a specific Target, that may be related to other similar Targets with the same TargetDefinition.

RESULTS: A new record is added to the collection of Targets in the system.

USE CASE TITLE Maintain TargetDefinition

DESC: Use a template to record information about what defines a Target.

RESULTS: A new template definition is added to collection of definitions.

USE CASE TITLE Maintain TaskDefinitions

DESC: Task Definitions are the templates used to describe tasks to be performed on Targets.

RESULTS: A new template definition is added to the collection of definitions.

USE CASE TITLE Maintain UsageTypes (see pages 9-10)

DESC: UsageTypes give context to Location definitions.

RESULTS: One or more UsageType(s) are created, updated, or deleted.

USE CASE TITLE Output Resource

DESC: Report on Resource collection.

RESULTS: A report.

USE CASE TITLE Output Target

DESC: Report on the Target collection.

RESULTS: A report.

USE CASE TITLE Output TargetDefinition

DESC: Report on Target Definitions

RESULTS: A report.

USE CASE TITLE Output TaskDefinition

DESC: Report on TaskDefinitions

RESULTS: A report.

USE CASE TITLE Print/Display JobTypes

DESC: Report on JobTypes

RESULTS: A report..

USE CASE TITLE Print/Display UsageTypes

DESC: Report on UsageTypes

RESULTS: A Report

SYSTEM FUNCTION GROUP: Job Creation

USE CASE TITLE Create AssessmentJob

DESC: Create an instance of a Job when the Target may or may not be known.

RESULTS: An assessment Job is added to the PendingJobOrders collection.

USE CASE TITLE Create DataCollectionJob

DESC: Create a Job to gather relevant information about a Target.

RESULTS: A Job will be added to the PendingJobOrders collection.

USE CASE TITLE Create MaintenanceJob

DESC: Create a non-system generated maintenance Job.

RESULTS: A non-system generated MaintenanceJob is added to the PendingJobOrders collection.

USE CASE TITLE CreateSystemPlannedJobs (see pages 11-13)

DESC: An operation to construct all System-Generated activities.

RESULTS: Puts newly system-created Jobs in the PendingJobOrders collection.

USE CASE TITLE Create ProjectJob

DESC: Define work to be done that may consist of one or more Jobs.

RESULTS: One or more Jobs, identified as being a part of a Project, are added to the PendingJobOrders collection.

USE CASE TITLE Create RepairJob

DESC: Create an Unplanned Event for some Target.

RESULTS: A RepairJob is added to the PendingJobOrders collection.

USE CASE TITLE Delete PendingJobOrder

DESC: Gives users the ability to delete instances of the PendingJob Orders collection.

RESULTS: A deleted PendingJobOrder.

USE CASE TITLE Modify PendJobOrdPriority

DESC: Gives users the ability to change the priority of a PendingJobOrder in order to change it's Scheduling position.

RESULTS: A Job that will be reviewed differently by the Scheduler.

SYSTEM FUNCTION GROUP: Job Scheduling

USE CASE TITLE Dispatch Job

DESC: A Job is physically delivered to the Resource it has been assigned to.

RESULTS: Mobile Worker receives an assignment.

USE CASE TITLE Add Resource to Job

DESC: This is used by Management to add an additional resource to a Job in a currently either in the PJA or DJA collections.

RESULTS: A resource/s will be added to the Crew on a Job.

USE CASE TITLE Delete PendingJobAssign

DESC: Gives users the ability to delete an instance of a PendingJobAssignment.

RESULTS: A deleted PendingJobAssignment.

USE CASE TITLE Schedule ShiftJobs

DESC: Invokes the Scheduler to create WorkSchedules for a supplied list of Sites and Resources for a shift.

RESULTS: WorkSchedules are created for all valid Resources.

SYSTEM FUNCTION GROUP: Job Scheduling

USE CASE TITLE ScheduleJob (see pages 14-15)

DESC: Invoke the Scheduler to assign a time and Resource for an instance of a PendingJobOrder that was added after a normal scheduling event.

RESULTS: A Job is added to the PendingJobAssignments collection.

SYSTEM FUNCTION GROUP: Job Execution

USE CASE TITLE Add Reference

DESC: Gives users the ability to indicate a second (or more) call for the same problem on an existing Job.

RESULTS: Updated Job record.

USE CASE TITLE Cancel Job

DESC: Gives users the ability to remove a Job that has been created for a Target that already has a Job on it for the same reason.

RESULTS: A deleted Job.

USE CASE TITLE Close Task

DESC: Allows the mobile worker to record all actions taken in a given task. Also records the time used to perform the task.

RESULTS: A closed Job, or display of the next task in sequence.

USE CASE TITLE Design Data Inquiry

DESC: Allows the Mobile Worker to search for design data, e.g., Make/Model/Serial, of a Target.

RESULTS: Pertinent Job data for the Mobile Worker.

USE CASE TITLE E-Mail

DESC: Allows the invocation of E-Mail services.

RESULTS: An Email is delivered.

USE CASE TITLE EndShift

DESC: Allows the mobile worker or other Resources to tell the system that it is time to go home.

RESULTS: A checked-out Resource.

USE CASE TITLE Get MoreJobs

DESC: The mobile worker will invoke this operation if he/she completes all Jobs on the current WorkSchedule.

RESULTS: Additional Jobs being added to the current WorkSchedules.

USE CASE TITLE Management Assignment

DESC: Allows management to override the Scheduler, e.g., if the Scheduler cannot find an acceptable Resource, one can be assigned by a manager.

RESULTS: In an update to a Resource's WorkSchedule and is transmitted to the handheld.

USE CASE TITLE Mgmt Reassignment

DESC: Allows management to remove an active JobAssignment from one Resource to another.

RESULTS: A Job removed from one Resource and given to another.

USE CASE TITLE Output TargetDefects

DESC: Gives the Mobile Worker the ability to display a list of defects that have been reported on a specific Target.

RESULTS: A report.

SYSTEM FUNCTION GROUP: Job Execution

USE CASE TITLE Pause Task

DESC: Allows the mobile worker to interrupt a Job in progress for some reason, which must be recorded. Any time elapsed during pause is added to the indirect time for that Resource
RESULTS: An interrupted Job.

USE CASE TITLE Related Target Inquiry

DESC: Allows the Mobile Worker to search for information about Targets that are related to the current Job
Target, e.g., electrical panel xx, or chilled water pumps, that are related to an Air-handler.
RESULTS: Useful Job information for the Mobile Worker.

USE CASE TITLE ReOpen AssessmentJob

DESC: Gives management the ability to reopen an AssessmentJob that was closed because the Resource dispatched by the System could not perform the assessment.
RESULTS: A new Resource would be manually attached to the AssessmentJob.

USE CASE TITLE Report Defects (see pages 16-18)

DESC: Gives the mobile worker the ability to record defects observed while working on a given Job.
RESULTS: Potentially a new Job will be created or, at least, pertinent data will be recorded.

USE CASE TITLE Resume Task

DESC: Allows the mobile worker to resume a Job at the point where it was paused.
RESULTS: A resumed Job.

USE CASE TITLE Review Created Job Order

DESC: Allows management to review new instances of CreatedJobOrders to validate or reject them.
RESULTS: A new PendingJobOrder that is allowed to be shown to the Scheduler, or an eliminated PendingJobOrder.

USE CASE TITLE Skip Task

DESC: Allows the mobile worker to not perform a given task. A reason must be given.
RESULTS: An assigned task not being completed, and additional time being added to the indirect labor record for this resource.

USE CASE TITLE Start Job

DESC: Allows the system to start collecting "direct" time for a given Job, and enables the display of the first task in sequence for that Job.
RESULTS: Clock is started and details are revealed in sequence.

USE CASE TITLE Start Shift

DESC: This operation is invoked at login time by a mobile worker at the beginning of a shift in order to download the current WorkSchedules for the Resources supported on a particular handheld device.
note: this operation also starts the clock to start tracking all
"direct" and "indirect" mobile worker time during a shift.
RESULTS: The mobile worker gets a current WorkSchedule.

USE CASE TITLE Target History Inquiry

DESC: Allows the Mobile Worker to search active JobHistory collection to learn what activities have been performed over some specified period of time.
RESULTS: A Target history is accessed.

USE CASE TITLE Update MobileWorker Stat

DESC: Gives users the ability to find out the current status of all or part of the Mobile Workforce.
RESULTS: A report.

SYSTEM FUNCTION GROUP: Management Reporting

***** NOTE: There are no diagrams included for this section. Reporting capabilities will be determined by the database platform(s) selected.**

USE CASE TITLE Job History Reports

DESC: Gives users the ability to create reports using various parameters, e.g., byType, byResource, etc., etc..

RESULTS: A report.

USE CASE TITLE Output DispJobAssignments

DESC: Report on the contents of the DispatchedJob Assignment collection.

RESULTS: A report.

USE CASE TITLE Output DispWorkSched

DESC: Report on the DispatchedWorkSchedules collection.

RESULTS: A report.

USE CASE TITLE Output JobInfo

DESC: Reports on specific instances of the Job collection.

RESULTS: A report.

USE CASE TITLE Output PendingJobAssign

DESC: Report on the contents of the PendingJob Assignments collection.

RESULTS: A report.

USE CASE TITLE Output PendingJobOrders

DESC: Report on the contents of the PendingJobOrder collection.

RESULTS: A report.

USE CASE TITLE Output PendWorkSched

DESC: Report in the contents of the Pending WorkSchedule collection.

RESULTS: A report.

USE CASE TITLE Output ResourceInfo

DESC: Gives management the ability to query the System for information about any Resource in the system. e.g., location, availability, etc..

RESULTS: Useful Management information about Resources.

USE CASE TITLE Output WorkSchedInfo

DESC: Reports on specific instances of the WorkSchedule collection.

RESULTS: A report.

USE CASE TITLE Policy Effectiveness

DESC: Gives management the ability to monitor the effectiveness of various Policy decisions, e.g., is a stated frequency on Air-handler maintenance producing the desired results.

RESULTS: A report.

USE CASE TITLE Resource Effectiveness

DESC: Allows management the ability to use various measures to determine the effectiveness, e.g., productivity measures.

RESULTS: A report.

USE CASE TITLE Target Effectiveness

DESC: Allows management to use various measures to validate the operating effectiveness of various Targets.

RESULTS: A report.

Use Case Schematic

Name: _____:Create_System_Planned_Jobs

Kind: _____:Method

Class _____:SysMaintJob

Description: _____:Create an instance of a System generated Planned Maintenance Job

Reads _____:Policy,Target,Jobtype, TaskDefinition

Supplied R:Resource, Initiator, CreateDate, T:TaskType,
J:JobType, P:Priority, TargetDates, L:Location, T:Target)

Changes _____:PendingJobOrders
new S:SysMaintJob

Sends _____:P:Planner : SysMaintJob_Created

Assumes _____:

Results _____:/* update as per CreateAssessmentJob */

IF

P:Priority and T:TargetDate Combination is valid according to Policy and Supplied

T:Target.Location =

Supplied L:Location and TLTaskType is valid for this SysMaintJob Job Type

THEN

New S:SysMaintJob with Status = Created and New TTL:TargetTaskList =

DefinedTasks where D:DefineTasks is for T:Target Supplied and TaskType =

Supplied,

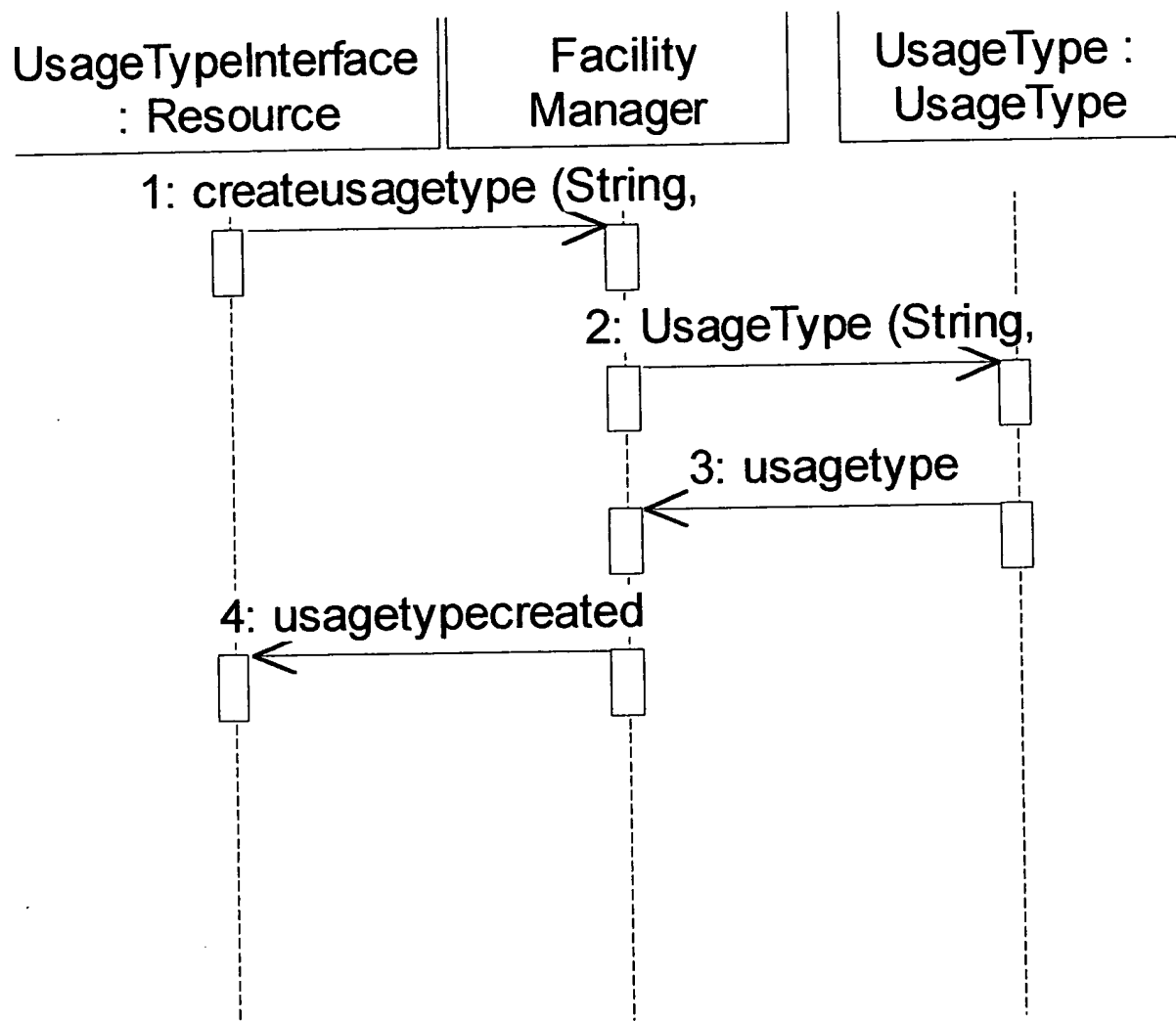
ADD

S:SysMaintJob to P:PJO

ELSE

Send Planner:SysMaintJob_Exception

Sequence Diagram for Create_Usage_Type



Use Case Schematic

Name: _____:Create_System_Planned_Jobs

Kind: _____:Method

Class _____:SysMaintJob

Description: _____:Create an instance of a System generated Planned Maintenance Job

Reads _____:Policy,Target,Jobtype, TaskDefinition

Supplied R:Resource, Initiator, CreateDate, T:TaskType,
J:JobType, P:Priority, TargetDates, L:Location, T:Target)

Changes _____:PendingJobOrders
new S:SysMaintJob

Sends _____:P:Planner : SysMaintJob_Created

Assumes _____:

Results _____:/* update as per CreateAssessmentJob */

IF

P:Priority and T:TargetDate Combination is valid according to Policy and Supplied

T:Target.Location =

Supplied L:Location and TLTaskType is valid for this SysMaintJob Job Type

THEN

New S:SysMaintJob with Status = Created and New TTL:TargetTaskList =

DefinedTasks where D:DefineTasks is for T:Target Supplied and TaskType =

Supplied,

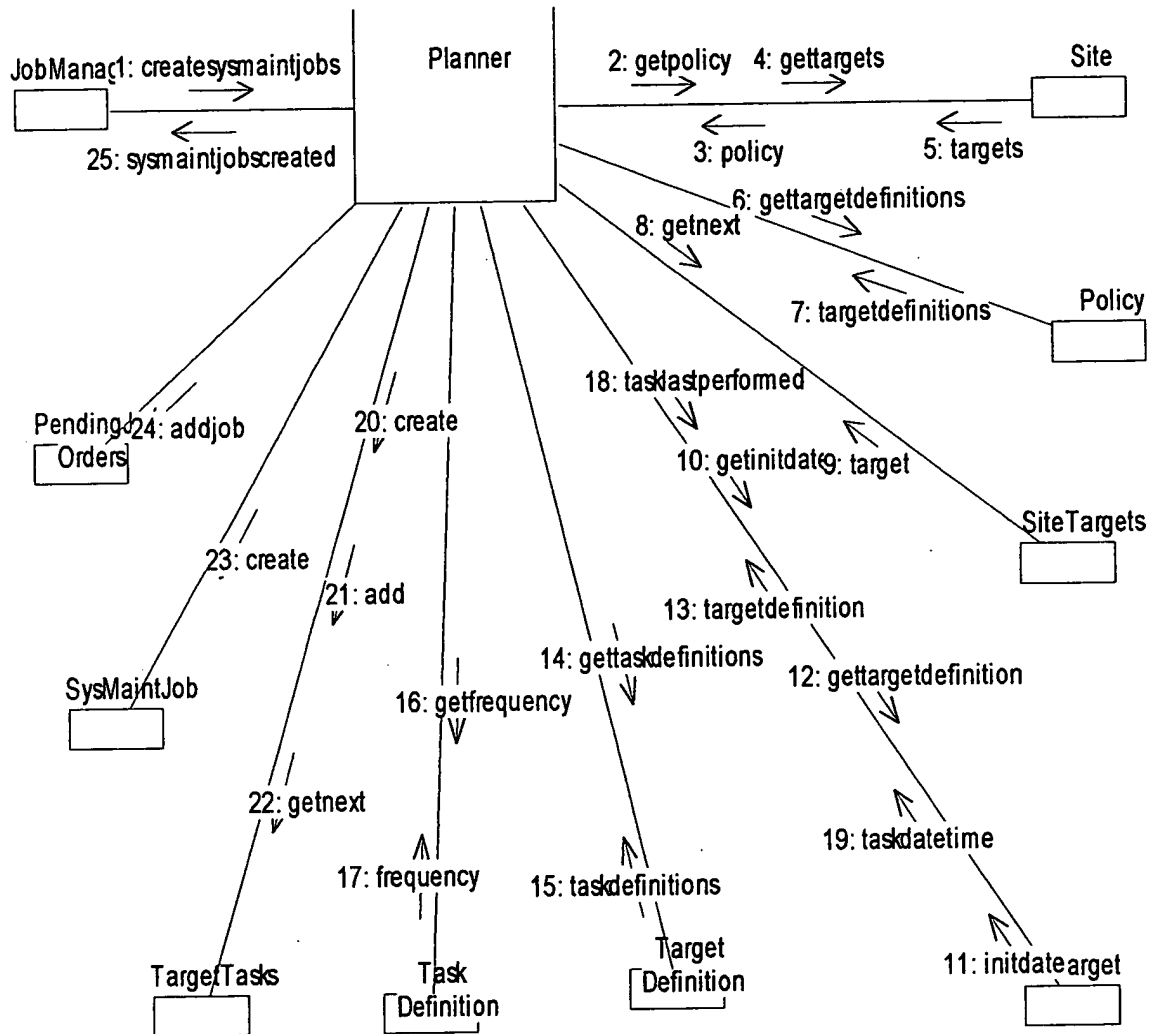
ADD

S:SysMaintJob to P:PJO

ELSE

Send Planner:SysMaintJob_Exception

Collaboration Diagram for Create_System_Planned_Jobs



Use Case Schematic

Name: _____:Schedule_Job

Kind: _____:Method

Class_____ :Scheduler

Description: _____:Invoke the Scheduler to assign a time and Resource for this new instance of a PJO
This updates an existing WorkSchedule.

Reads _____:Supplied (JobID, Resources, DecisionFactors)
Clock, PJA, PJO, Job, Resource, ResourceSchedule

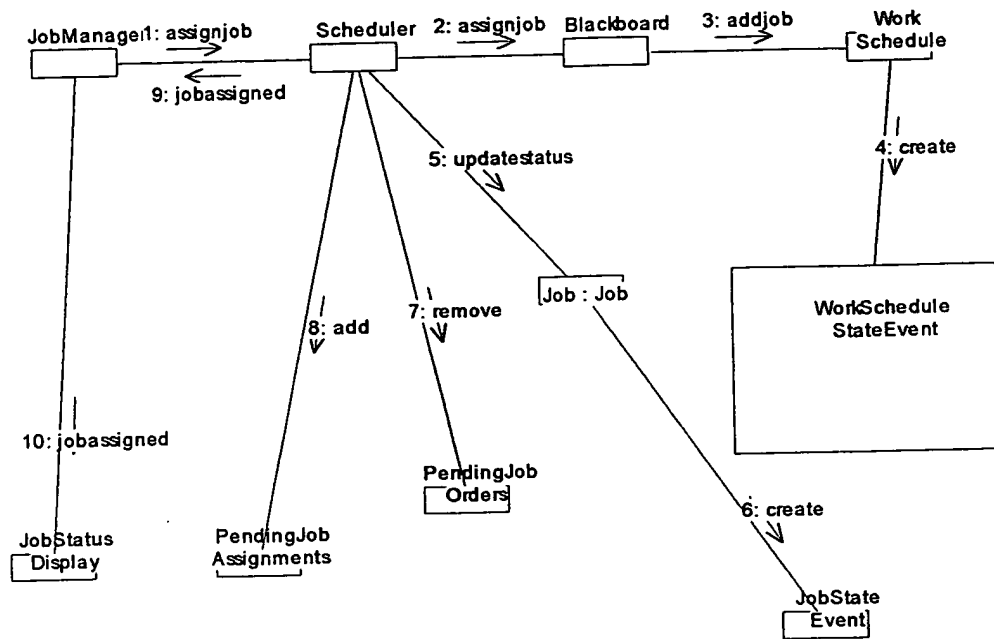
Changes _____:Job, WorkSchedule, PJA, PJO, PWS

Sends _____: Policy/MGR:Job_Assigned(WorkSchedule/s Ids)
Policy/MGR:Job_Unassigned (reason)
Policy/MGR:Scheduler_Exception

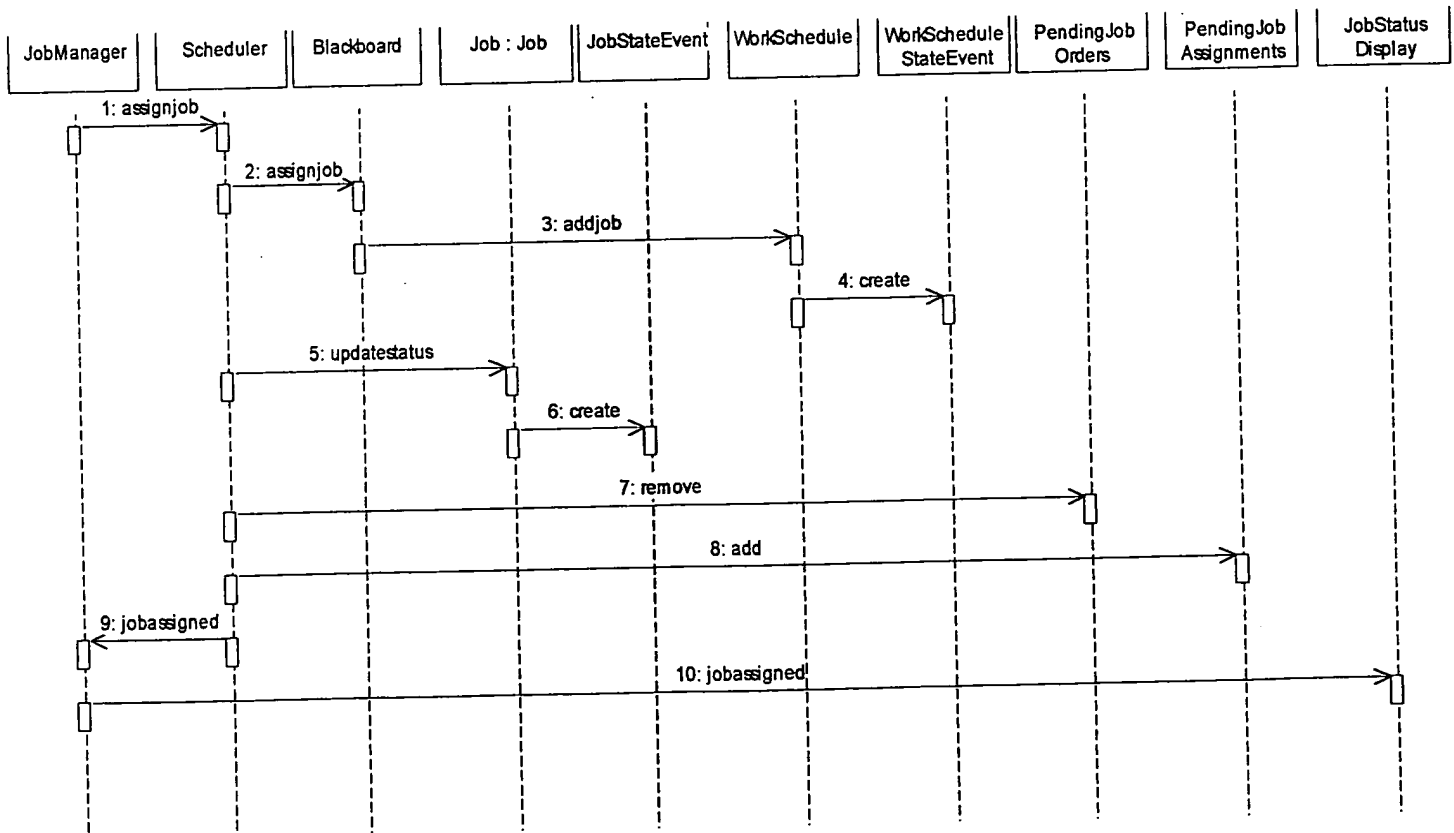
Assumes _____:Job identified is valid and in the PJO collection and Policy has determined
to assign this Job NOW. Policy has identified candidate Resources, and
DecisionFactors.

Results _____:IF Resources and DecisionFactors Found
THEN
(Collect decision Info)
FOR Each Resource (Get Decision Factors)
IF Current WorkSchedule on DWS
THEN
Update DWS WorkSchedule (HH to Server)
(This is a passive pull of the HH Resident Current WorkSchedule for this
Resource)
Read CurrentWorkSchedule for:
Current Location
Current JobStatus
Current JobPriority
Est. Time of Current Job
Next JobPriority
Next JobLocation
Read ResourceSchedule for;
Est. Time Remaining on Shift (May be evident on WS)
END
MakeDecision (Based on factors)
Research rules-based AI logic....
IF Job_Assigned
THEN
Returns Updated: (J:Job, WorkSchedules, PJA, send Policy/MGR/Dispatcher:
Job_Assigned()
ELSE
Returns send Policy/MGR:Job_Unassigned
ELSE send Policy/MGR:Scheduler_Exception
Trigger Exception

Collaboration Diagram for Schedule_Job_Operation



Sequence Diagram for Schedule_Job_Operation



Use Case Schematic

Name: _____ :MWReportDefect

ID: _____

Kind: _____ :Operation

Class: _____ :Target

Description: This operatin is invoked by a MW resource in order to record a defect or problem about _____ a Target. This informatin is reviewed by a MGR and it could lead to a project job being created.

Reads: _____ :

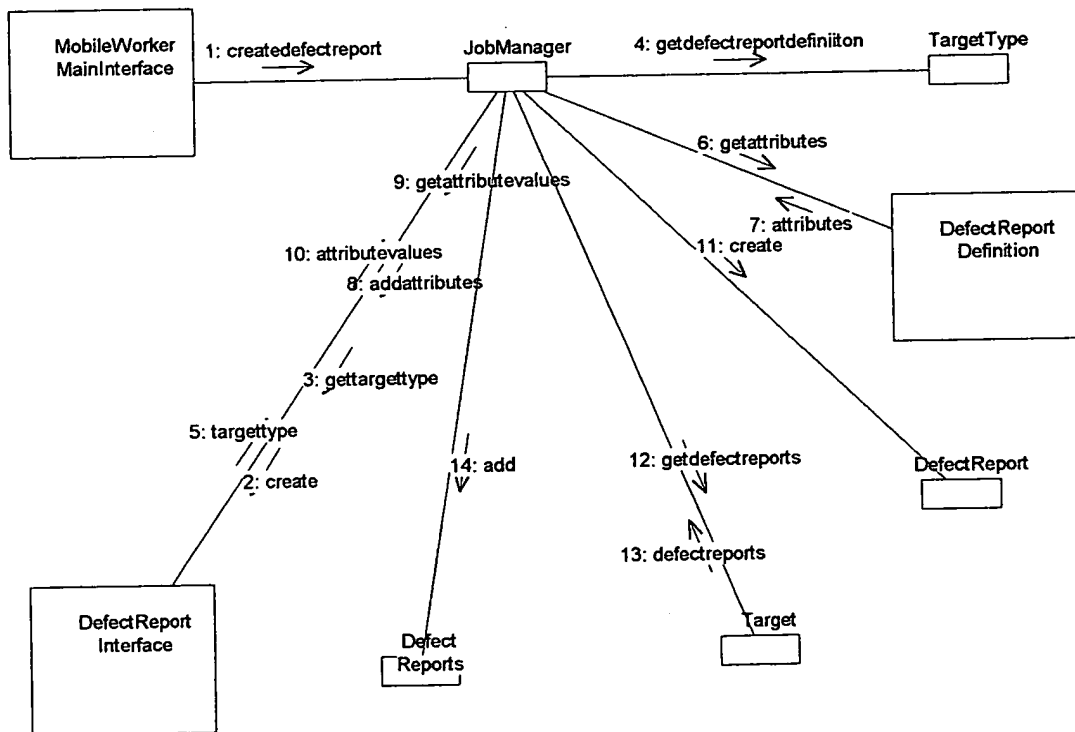
Changes: : new TargetDefect,
 Target.Defects (collection)

Sends _____ :Resource:TargetDefectCreated,
 Resource:Server_Unavailable

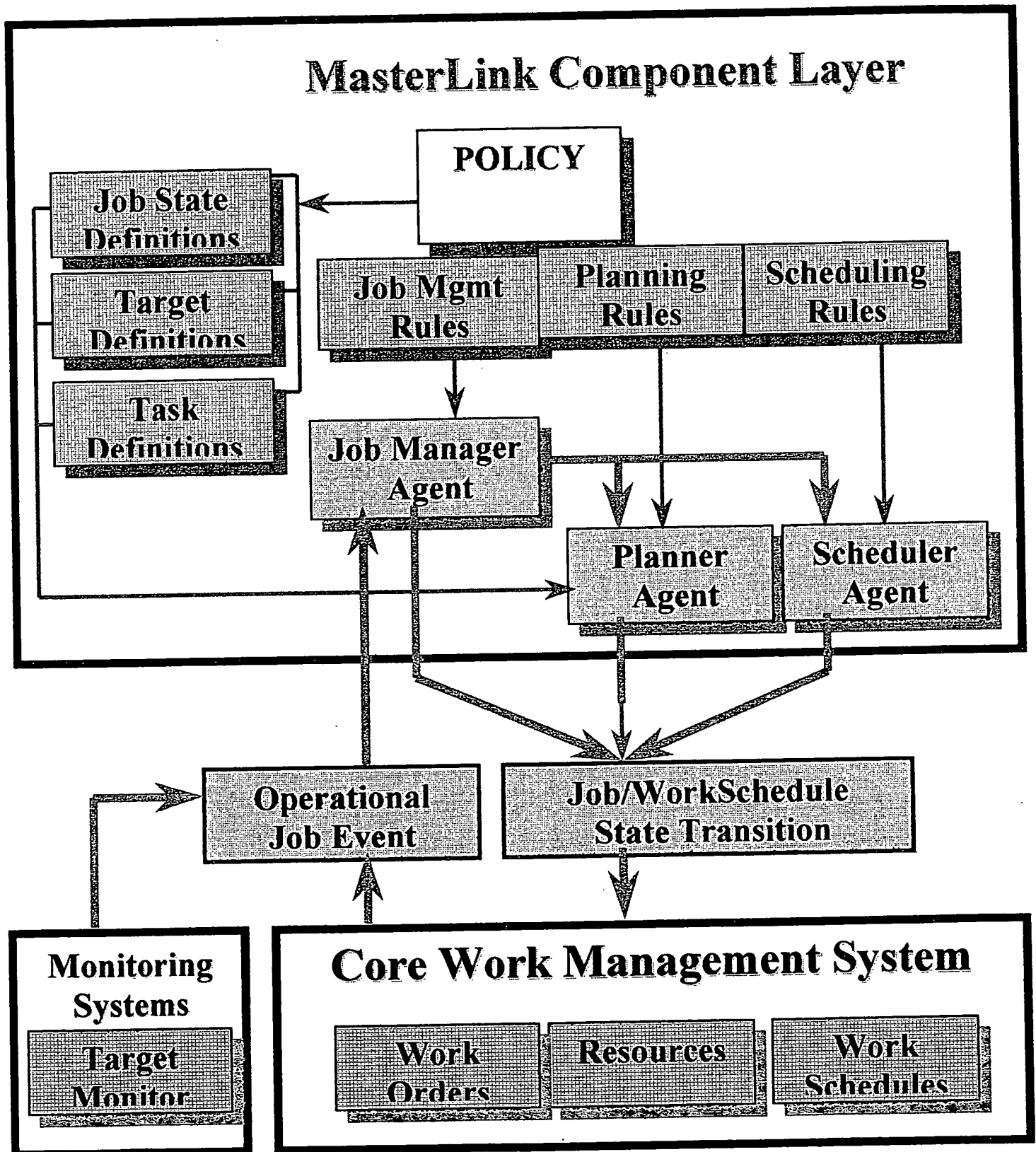
Assumes _____ :Resource is valid, and is on shift

Results: _____ :/* Depending on whether you are entering a defect for a Target that is on a Job in the MW WorkSchedule, or a Target unrelated to any Job and therefore probably not local to the handheld, this may be different */
 Get Target ID
 New TargetDefectDisplay(TargetID)
 /* are defects the same for all target types? */
 Get TargetDefect

Collaboration Diagram for Report_Defects



System Architecture Overview

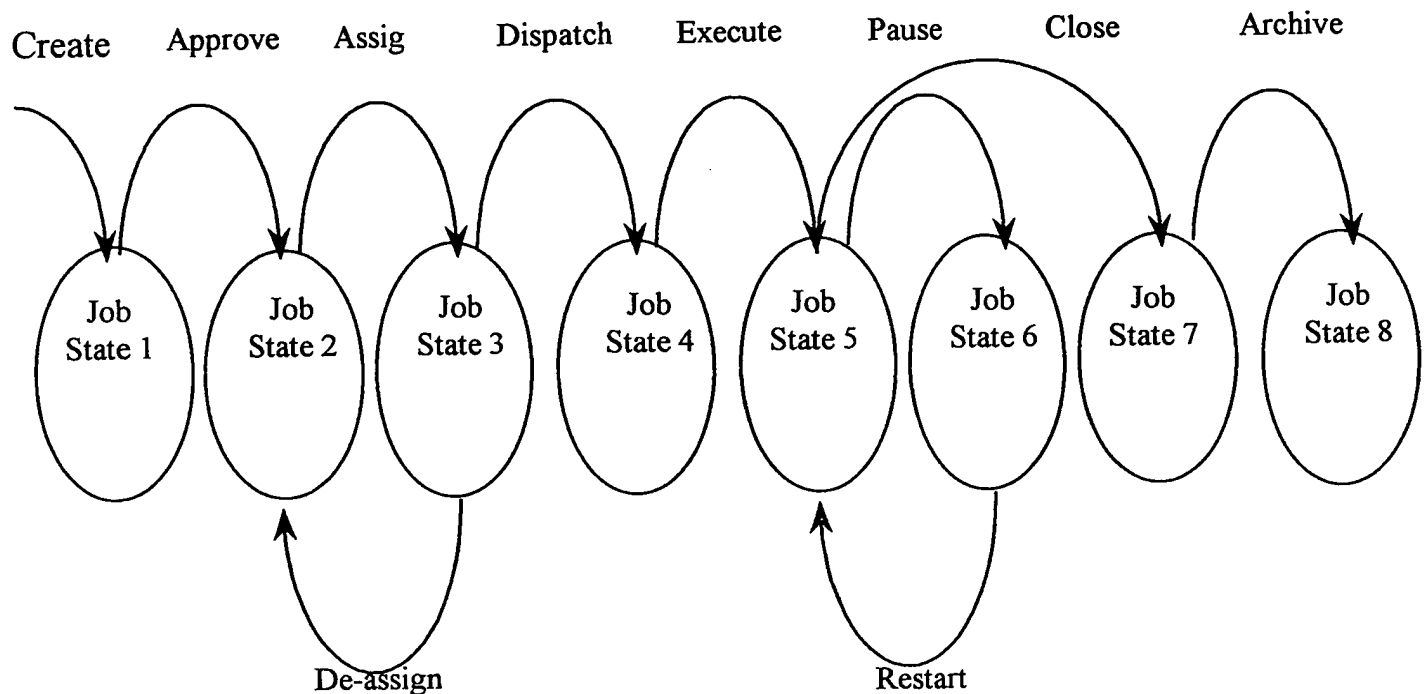


Physical Architecture for MasterLink's Distributed Intelligent Work Management System

Vertical Domain Work Management Application	The domain specific objects involved in a physical implementation of a workflow management application based on the MasterLink framework. This includes such things as work management policies, work targets in a classification hierarchy, rules for MasterLink agents, task definitions, job types, job state transitions, and work schedule state transitions. These are specified on a case by case basis as part of the system initialization process, e.g. for a facilities maintenance domain, a home health care domain, etc.
MasterLink Collaborative Agent Framework	The MasterLink collaborative agents, and the framework of classes supporting these agents provide the basis for the intelligent work management application. The relationships between the domain specific objects referred to above and the intelligent work management agents are defined in this framework.
Inference/Constraint Engine Libraries	The MasterLink agents are implemented as classes that are derived from commercially marketed artificial intelligence products. The ability of a MasterLink agent to use a set of rules or constraints to make a workflow management decision is based on this technology.
Object Request Broker	The mechanism by which distributed instances of application objects can send messages to each other. The support for these distributed objects to communicate over a wireless connection is evolving. Until mature, existing wireless protocols may have to be implemented.
Class Libraries	Depending on the language, these are commercially available libraries for common programming functions, such as file i/o, directory services, string handling, date/time functions, and database connectivity.
Language Compiler	The programming language used to implement the application. At this point in time the distributed object oriented options include C++ and Java. This is due to the compatibility requirement with the AI products, Orbs and Databases.
Operating System	The operating system which must be capable of supporting the language and other off the shelf components mentioned above. Typically this is Unix or NT on servers, and clients will vary depending on their type, e.g. a desktop LAN connected client versus a handheld wireless network device.

JOB STATE TRANSITIONS

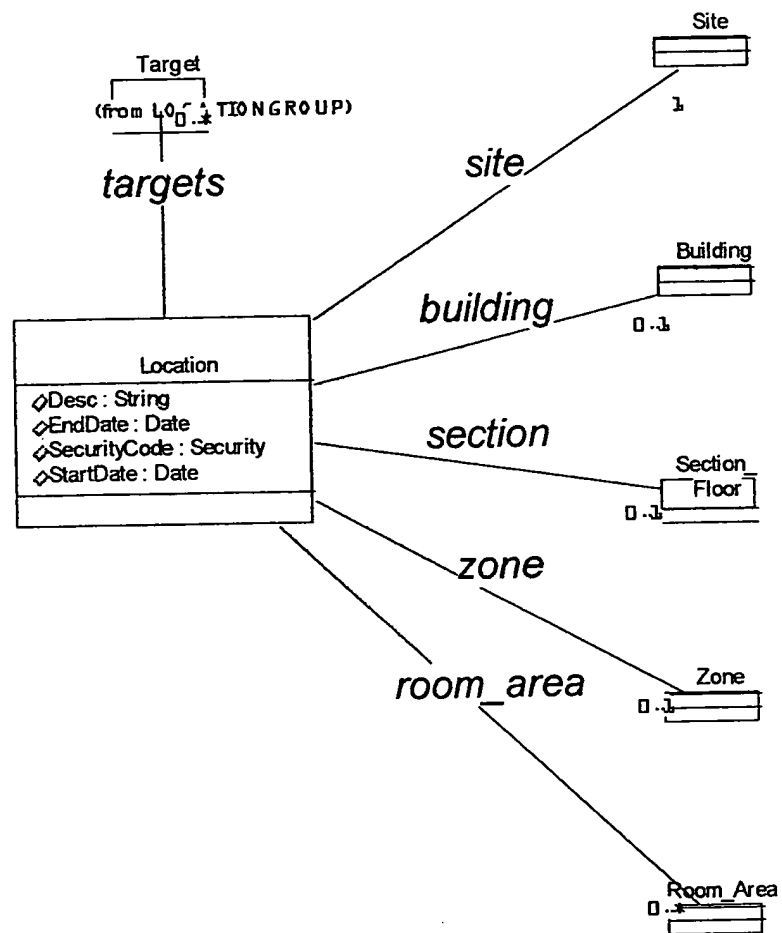
Example Job Type



- A series of states and transitions will be defined for each “type” of job to be managed by the system.
- A set of business rules governing each possible transition will be determined. Analysis will include consideration for vertical domain classes.
-
- System agents will use sets of rules to automate selected transitions. External interfaces will support manual transitions and overrides.
- “Planner” agent will address the generation or creation of jobs containing planned tasks.
- “Scheduler” agent will address the assignment of jobs to resources and time.
- “Dispatcher” agent will handle delivery of work schedules to resources.
- The worker, through a mobile device interface, will be the source for many transitions.
- “Job Manager” agent will act as a communications traffic cop receiving messages, representing events, from the external interfaces (either GUI or system based) , from the internal system agents. and from other MasterLink internal classes.

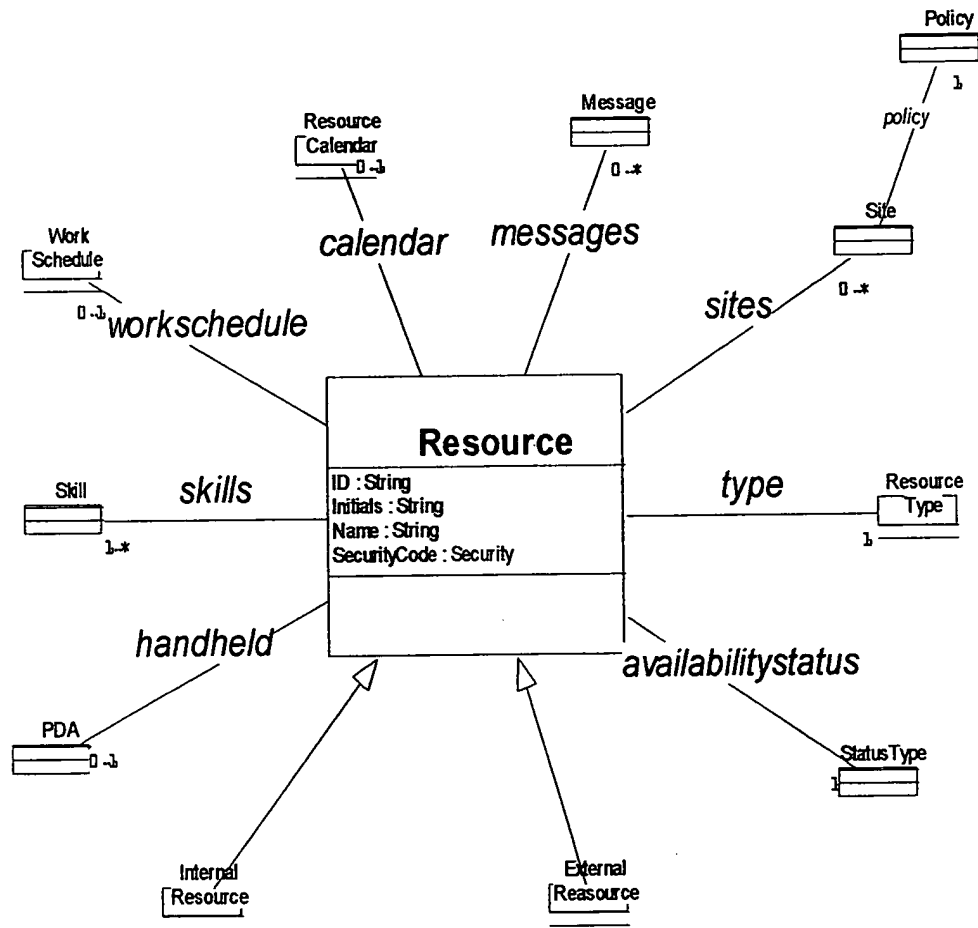
DOMAIN MODELS

LOCATION CLASS



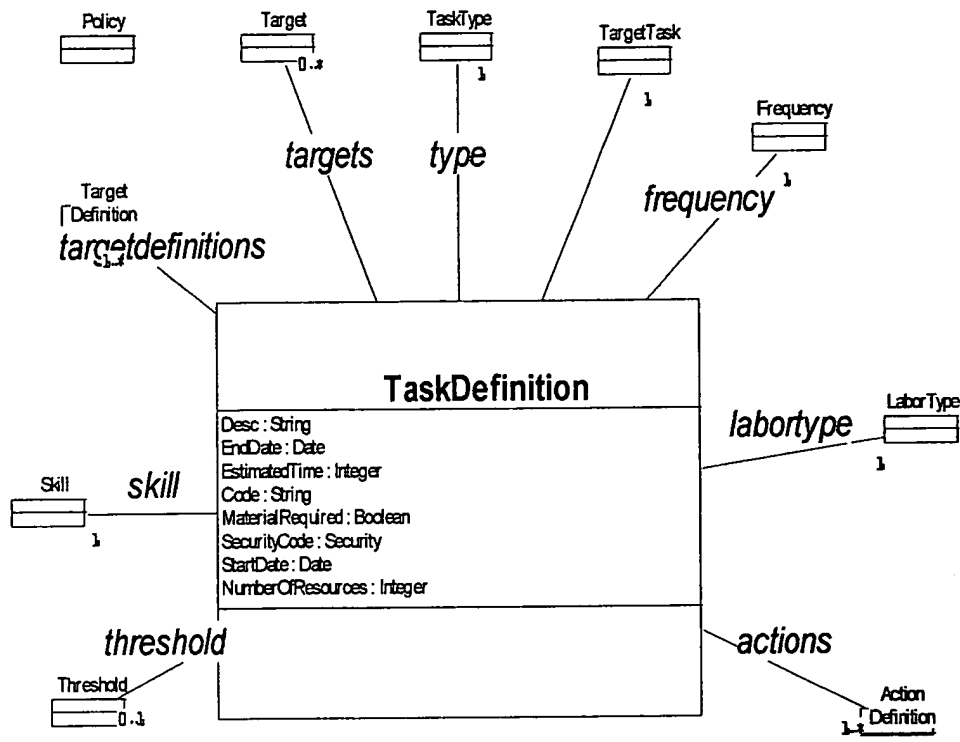
DOMAIN MODELS

RESOURCE CLASS



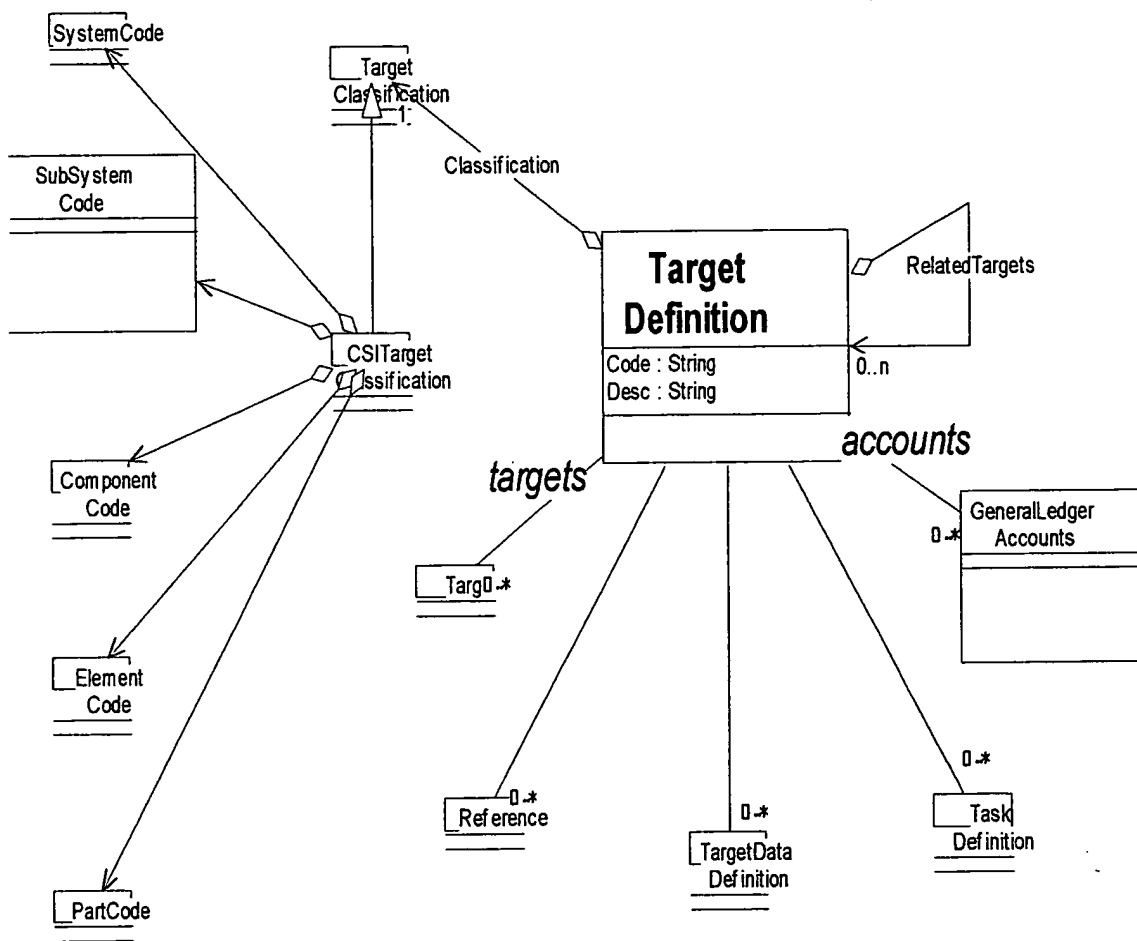
DOMAIN MODELS

TASKDEFINITION CLASS



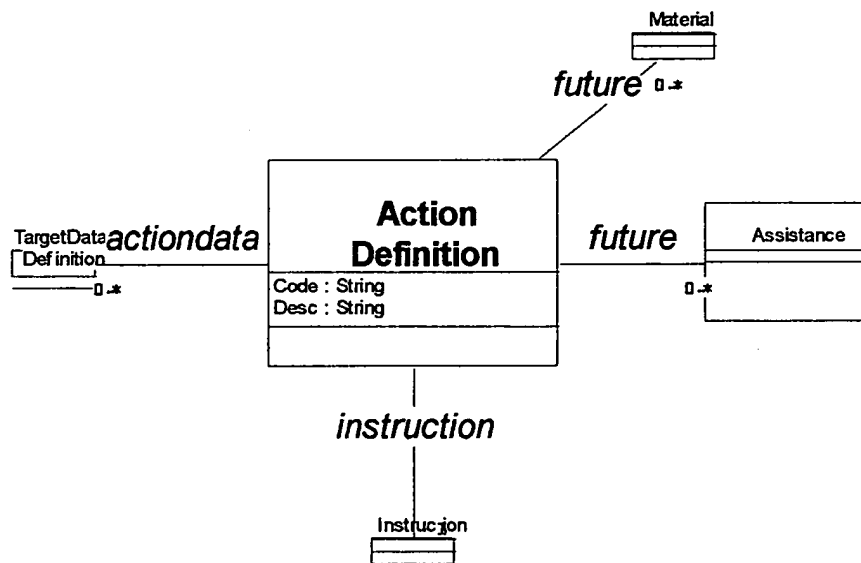
DOMAIN MODELS

TARGETDEFINITION CLASS



DOMAIN MODELS

ACTIONDEFINITION CLASS



DOMAIN MODELS

POLICY CLASS

